REMARKS

Claims 1-41 are pending. In view of the following remarks, Applicant respectfully solicits allowance of the subject application and furtherance onto issuance.

References

The Office requested the references "Windows 95 WIN32 Programming API Bible" by Richard Simon and "Inside OLE, Second Edition" by Kraig Brockschmidt. Accordingly, the requested references are being supplied herewith.

35 U.S.C. §103(a)

Claims 1-3, 6-17, 19, 22, 23, 29-31 and 36-41 are rejected under 35 U.S.C. §103(a) as being unpatentable by U.S. Patent No. 5,379,432 to Orton et al. (hereinafter, "Orton") in view of Understanding ActiveX and OLE to David Chappell, pages 4-5 (hereinafter, "Chappell"). Applicant respectfully traverses the rejections.

Claim 1 is directed to a method of factoring operating system functions that includes defining criteria that governs how functions of an operating system are to be factored into one or more groups. The functions are factored into one or more groups based upon the criteria. Groups of functions are associated with programming objects that have data and methods. The methods correspond to the operating system functions and are effective to provide an object oriented operating system. The programming objects are configured to be instantiated throughout a remote computing system.

Claim 14 recites a method of factoring operating system functions. A plurality of operating system functions that are used in connection with operating system resources are factored into first groups based upon first criteria. The first groups are also factored into individual sub-groups based upon second criteria. Each sub-group is assigned to its own programming object interface. A programming object interface represents a particular object's implementation of its collective methods that are effective to provide an object-oriented operating system. Individual objects having associated programming object interfaces are configured to be instantiated throughout a remote computing system.

The Office first asserts Orton at Col. 7, lines 1-67, Col. 8, Lines 1-14, and figure 4 for defining criteria that governs how functions of an operating system are to be factored and associating groups of functions with programming objects. The Applicant disagrees. The Office then correctly asserts that Orton "is not explicit with reference to the programming objects being configured to be instantiated throughout a remote computing system." *Office Action Dated March 11, 2004, Page 3.* The Applicant agrees, Orton does not disclose, teach or suggest "programming objects being configured to be instantiated throughout a remote computing system" as claimed in Claim 1.

The Office then asserts Chappell to correct the defects of Orton and asserts that "[o]ne would have been motivated to make such a modification to provide one application's services to another (page 4 lines 22 – 35)." *Office Action Dated March 11, 2004, Page 3.* The Applicant respectfully disagrees. It is respectfully submitted that the Office has engaged in impermissible hindsight reconstruction in rejecting the claims because none of the submitted references, alone or in combination, contain the motivation or suggestion for the asserted combination.

Additionally, the asserted combination of Orton and Chappell does not disclose, teach or suggest the methods of claims 1 and 14.

To establish a prima facie case of obviousness, three basic criteria must be met. First there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one or ordinary skill in the art, to modify the reference or to combine reference teachings. Second, there must be a reasonable expectation of success. Finally, the prior art reference (or references when combined) must teach or suggest all the claim limitations. The teaching or suggestion to make the claimed combination and the reasonable expectation of success must both be found in the prior art and not based on applicant's disclosure. *See MPEP 706.02(j)* and *In re Vaeck*, 947 F.2d 488, 20 USPQ2d 1438 (Fed. Cir. 1991). Thus, obviousness cannot be established by combining the teaching of the prior art to produce the claimed invention, absent some teaching or suggestion supporting the combination. The Office may not use the patent application as a basis for the motivation to combine or modify the prior art to arrive at the claimed invention.

Beginning at page 13 of the subject specification, an exemplary distribution is described of an operating system's resources across one process boundary and one machine boundary in a distributed computing system. In the described example, in relation to FIG. 4, resource object 48 is instantiated in-process (i.e. inside the application's process), resource object 50 is instantiated in another process on the same machine (i.e. local), and resource object 52 is instantiated on another machine (i.e. remote). Use of remote and local resources is described in an additional embodiment beginning at page 22 of the subject specification. An operating system is provided with the ability to determine, based on the specified unique identifier, whether it has the resource that is requested. If it does not, the operating system can ascertain the location of the particular resource and retrieve it so that the application can have the requested resource. The location from which the resource is retrieved can be across process and machine boundaries. As an example, consider the following. If an application asks for a specific version of a "ReadFile" interface, and the operating system does not support that version, the operating system may know where to go in order to download the code to implement the requested functionality. Software code for the specific requested interface may, for example, be located on a web site to which the operating system has access. The operating system can then simply access the web site, download the code, and provide the resource to the application.

Orton is directed to an object-oriented interface for a procedural operating system in the "context of executing the object oriented application 103A on the computer platform 102." *Orton, Col. 8, Lines 21-22.* Orton describes a limited functionality procedural operating system, such as the Mach micro-kernel. *Orton, Col. 5, Line 67 to Col. 6., Line 2.* The executable entity in Mach is known as a thread. *Orton, Col. 11, Line 25.* Threads use ports to communicate with each other. A port is "basically a message queue inside the kernel if [sic] at threads can add messages to or remove message from, if they have the proper permissions to do so." *Orton, Col. 15, Lines 39-42.* Ports exist solely in the kernel and can only be manipulated via port rights. *Orton, Col 15, Lines 48-49.* Thus, Orton describes that messages are utilized by threads to communicate with each other through the use of ports. Nowhere in Orton is remote communication even mentioned. This is further supported by the Summary of the Invention in Orton, which is excerpted as follows:

> The present invention is directed to a system and method of enabling an object-oriented application to access in an object-oriented manner a procedural operating system having a native procedure interface. ***The system includes a computer and a memory component in the computer.*** *Orton, Col. 3, Lines 50-55 (emphasis added).*

Therefore, the system described in Orton is implemented in a single computer. Indeed, nowhere in Orton is remote communication or a machine boundary even discussed. Rather, Orton describes a computer platform, such as "an International Business Machines (IBM) computer or an IBM-compatible computer .... [or an] Apple computer". *Orton, Col. 6, Lines 6-10.*

The Office then asserts Chappell for disclosure of a remote computing system and as motivation for a combination of Chappell and Orton. Specifically, the Office asserts in the *Office Action Dated March 11, 2004, Page 3* that "[o]ne would have been motivated to make such a modification to provide one application's services to another (page 4 lines 22 – 35)." Assuming that the Office is referring to page 4 of Chappell, the referenced portion of Chappell is excerpted as follows:

> All OLE technologies and all the ActiveX technologies described in this book are built on the foundation provided by COM. So just what is COM? To answer this question, think first about another: how should on chunk of software access the service provided by another chunk of software? Today, as shown in Figure 1-2, the answer depends on what those chunks of software are. An application might, for example, link to a library and then access the library's services by calling the function in the library. Or one application might use the service provided by another, which runs in an entirely separate process. In this case, the two local processes typically communicate by using an interprocess communication mechanism, which usually requires defining a *protocol* between the two applications (a set of message allowing one application to specify its request and the other to respond appropriately). A third example is an application that

> might use a service provided by an operating system. Here the application commonly makes system calls, each of which is handled by the operating system. Or, finally, an application might need the services of software that is running on a completely different machine, accessible via a network. Many different approaches can be used to access these services, such as ***exchanging messages with the remote application*** or issuing remote procedure calls. *Chappell, Page 4, Line 23 to Page 5, Line 9 (emphasis added).*

As shown in the above excerpt, although Chappell describes that an application might need the services of software that is running on a completely different machine, Chappell does not include motivation or suggestion for associating groups of ***operating system functions*** with programming object being configured to be instantiated through a remote computing system. Rather, Chappell describes "exchanging messages ***with the remote application***" and makes no mention of operating system functions associated with programming objects that are configured to be instantiated throughout a remote computing system. *Chappell, Page 5, Lines 8-9 (emphasis added).*

Chappell, for instance, states that "an application might need the services of software that is running on a completely different machine, accessible via a network", but the referenced software is a remote application as shown in the above excerpt, e.g., "exchanging messages with ***the*** remote application". *Chappell, Page 5, Line 5-9 (emphasis added).* Thus, the section asserted by the Office describes making one application's services available to another application, and does not describe, teach or suggest ***operating system functions*** configured to be instantiated throughout a remote computing system. This is further supported in the assertion made by the Office, "[o]ne would have been motivated to make such a modification to provide ***one application's services to another*** (page 4 lines 22 – 35)." *Office Action Dated March 11, 2004, Page 3* *(emphasis added)*. It is respectfully submitted that Chappell is relied on solely for support of a remote computing system, and does not include motivation or suggestion for being combined with Orton as asserted by the Office. Absent the present application, the Orton reference describes an object-oriented interface for a procedural operating system, while Chappell merely describes the use of COM objects for communication between applications. Even assuming solely for the sake of argument that Orton and Chappell are combinable as asserted by the Office, the combination does not disclose, teach or suggest the methods of claims 1 and 14 as previously described. Accordingly, for at least these reasons, claims 1 and 14 are allowable.

Claims 2-13 depend either directly or indirectly from claim 1 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 1, are neither shown nor suggested in the references of record, either singly or in combination with one another. For example, claim 7 recites "instantiating a plurality of programming objects across a machine boundary". Orton does not disclose, teach or suggest the presence of a machine boundary nor even mention more than one computer platform. Chappell does not cure the defects of Orton as previously described. Therefore, for at least this reason, claim 7 is allowable.

Claims 15-23 depend either directly or indirectly from claim 14 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 14, are neither shown nor suggested in the references of record, either singly or in combination with one another.

Claim 29 recites an operating system application program interface embodied on a computer-readable medium comprising a plurality of object interfaces. Each object interface is recited to be associated with an object that includes one or more methods that are associated with and can call functions of an operating system that does not comprise the object interfaces. At least one object is configured to be remotely instantiated.

The Office correctly asserts that Orton "is not explicit with reference to at least one the objects being configured to be remotely instantiated." *Office Action Dated March 11, 2004, Page 7.* The Office then asserts Chappell to correct the defects of Orton. As previously described, however, nowhere does Chappell disclose or even suggest any such subject matter. Chappell merely describes communication between applications, while Orton describes an operating system on a single computing device. It is respectfully submitted that the assertion of Chappell at Page 4, Lines 22-35 by the Office does not supply any suggestion or motivation for forming the asserted combination with Orton as previously described. As excerpted above, the referenced portion of Chappell merely describes exchanging messages with a remote application. Thus, Chappell does not correct the defects of Orton, alone or in combination. Chappell does not include suggestion or motivation for being combined with Orton nor does the combination disclose, teach, or suggest the operating system application programming interface of claim 29. Accordingly, for at least these reasons this claim is allowable.

Claims 30-35 depend either directly or indirectly from claim 29 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 29, are neither shown nor suggested in the references of record, either singly or in combination with one another.

Claim 36 recites an operating system that includes a plurality of programming objects having interfaces. The programming objects represent operating system resources, and the interfaces define methods that are organized in accordance with whether they create an operating system resource or not. The programming objects are configured to be called either directly or indirectly by an application. The methods are configured to call operating system functions responsive to being called directly or indirectly by an application. The programming objects are configured to be instantiated throughout a remote computing system.

Again, the Office correctly asserts that Orton "is not explicit with reference to at least one the objects being configured to be remotely instantiated." *Office Action Dated March 11, 2004, Page 8.* Orton neither discloses nor suggests "programming objects being configured to be instantiated throughout a remote computing system" as claimed in claim 36. As previously described, Chappell does not correct the defects of Orton. Chappell does not include suggestion or motivation for being combined with Orton nor does the combination disclose, teach, or suggest the operating system of claim 36. Accordingly, for at least this reason, claim 36 is allowable.

Claims 37-40 depend from claim 36 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 36, are neither shown nor suggested in the references of record, either singly or in combination with one another.

Claim 41 recites a method of converting an operating system from a non-object-oriented format to an object oriented formal. The operating system includes a plurality of operating system functions that are callable to create or use operating system resources. A plurality of programming object interfaces are defined that define methods that correspond to the operating system functions. Programming objects that support the interfaces are callable either directly by an application that makes object-oriented calls, or indirectly by an application that makes function calls. The programming objects being configured to be instantiated throughout a remote computing system. A programming object interface is called either directly via an object-oriented call, or indirectly via an indirection that transforms a function call into an object-oriented call. Responsive to the calling, an operating system function is called with a method of the programming object that supports said programming object interface.

The Office correctly asserts that Orton "is not explicit with reference to at least one the objects being configured to be remotely instantiated." *Office Action Dated March 11, 2004, Page 9.* Orton neither discloses nor suggests "programming objects being configured to be instantiated throughout a remote computing system" as claimed in claim 41. Chappell does not correct the defects of Orton. Chappell describes communication with a remote application, and not programming object interfaces that define methods that correspond to the ***operating system functions*** and that are configured to be instantiated throughout a remote computing system. Accordingly, for at least these reasons, claim 41 is allowable.

For at least these reasons, claims 1-3, 6-17, 19, 22, 23, 29-31 and 36-41 are allowable over Orton in view of Chappell. Applicant respectfully requests that the §103 rejection of claims 1-3, 6-17, 19, 22, 23, 29-31 and 36-41 be withdrawn.

35 U.S.C. §103(a)

Claims 4, 5, 18, 20, 21, 24-28 and 32-35 are rejected under 35 U.S.C. §103(a) as being unpatentable over Orton in view Chappell and further in view of U.S. Patent Number 6,334,157 to Oppermann et al. (hereinafter, "Oppermann"). Applicant respectfully traverses the rejections.

Claim 24 recites a method of factoring operating system functions which includes factoring a plurality of operating system functions into interface groups based upon the resources with which a function is associated. The interface groups are factored into interface sub-groups based upon each function's use of a handle that represents a resource. The interface sub-groups are organized so that at least one of the interface sub-groups inherits from at least one other of the interface sub-groups. Individual interface sub-groups are associated with individual programming objects that can be instantiated throughout a remote computing system.

The Office correctly asserts that Orton "is not explicit with reference to at least one the objects being configured to be remotely instantiated." *Office Action Dated March 11, 2004, page 12.* Neither Orton nor Chappell, alone or in combination, disclose or suggest a method that factors operating system functions where "individual objects sub-groups being associated with individual programming objects that can be instantiated throughout a remote computing system" as claimed in claim 24. Chappell does not correct the defects of Orton.

Additionally, Oppermann, alone or in combination with any of the submitted references, does not correct these defects.

The Office asserts Oppermann to describe factoring groups into interface sub-groups based upon each function's handle. In making out the rejection, the Office contends that Oppermann meets the subject matter of this claim. Specifically, the Office argues that Oppermann teaches a plurality of operating system functions, a handle, a resource, and organizing the interface sub-groups so that at least one of the interface sub-groups inherits from at least one other of the interface sub-groups. Applicant respectfully disagrees with the Office's interpretation of this reference and traverses the rejection.

Specifically, claim 24 recites a step in which the interface groups are factored into interface sub-groups ***based upon each function's use of a handle*** that represents a resource. Nowhere does Oppermann teach or even suggest this feature. The Office asserts Oppermann at Col. 8, Lines 16-67 and at Col. 21, Lines 35-51, which are excerpted as follows:

> As part of the preferred architecture, an application program supports the IAccessible interface, which allows clients to reap the benefits of using the preferred architecture. As shown in FIG. 3, a client 302, such as an accessibility aid, accesses a server 304, such as an application program with user interface elements, via the IAccessible interface 306. The IAccessible interface 306 contains function members which allow the client 302 to directly access and manipulate the implementation of the user interface elements of the server 304. These function members allow the client 302 to access the user interface elements in the following ways: by navigating through the user interface elements, by retrieving the name for each user interface element, by determining whether each user interface element is visible or hidden, by retrieving the text description for each user interface element, by identifying the location of the user interface element, by determining the parent or child of a user interface element,

> and by determining the value of the user interface element. Some user interface elements have an associated value. For example, a clock user interface element has a value of the current time. *Opermann, Col. 8, lines 16-67.*
>
> The LResultFromObject Function
>
> The LResultFromObject function receives an identifier of an interface and returns an LResult. The LResultFromObject function is defined below.
>
> LRESULT LresultFromObject(REFIID riid, WPARAM wParam, LPUNKNOWN punk);
>
> This function receives a pointer to an IUnknown interface for a user interface element and generates a handle for the pointer, known as an LRESULT value. This function returns a positive number if successful or an error code otherwise.
>
> riid--This parameter is a reference identifier of the interface that will be provided to the client. For example, the riid parameter may identify the IAccessible object interface.
>
> wparam--This parameter is a value as provided in the wParam parameter received with the associated WM_GetObject message. *Opermann, Col. 21, lines 35-51.*

Thus, Oppermann simply describes the use of a handle. Oppermann neither discloses nor suggests factoring interface groups into interface sub-groups ***based upon each function's use of a handle*** that represents a resource. Accordingly, for at least these reasons, this claim is allowable.

Claims 25-28 depend either directly or indirectly from claim 24 and are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features which, in combination with those recited in claim 24, are neither shown nor suggested in the references of record, either singly or in combination with one another.

Claims 4, 5, 18, 20, 21 and **32-35** depend either directly or indirectly from allowable independent claims are allowable as depending from an allowable base claim. These claims are also allowable for their own recited features. For example, claims 4 and 5 recite "creating a hierarchy of object interfaces in which certain interfaces can inherit from other interfaces" and "creating a hierarchy of object interfaces in which certain interfaces can aggregate with other interfaces", respectively, which are neither shown nor suggested in the references of record, either singly or in combination with one another.

For at least these reasons, claims 4, 5, 18, 20, 21, 24-28 and 32-35 are allowable over Orton in view of Chappell and further in view of Oppermann. Applicant respectfully requests that the §103 rejection of claims 4, 5, 18, 20, 21, 24-28 and 32-35 be withdrawn.

Conclusion

All pending claims 1-41 are in condition for allowance. Applicant respectfully requests reconsideration and prompt issuance of the subject application. If any issues remain that prevent issuance of this application, the Examiner is urged to contact the undersigned attorney before issuing a subsequent Action.

Respectfully Submitted,

Dated: June 10, 2004

By: [illegible]

William J. Breen, III
Reg. No. 45,313
(509) 324-9256 x249